Using ZeroFault to Find Memory Errors and Leaks in Large Parallel AIX Applications

John Gyllenhaal

Livermore Computing

Lawrence Livermore National Laboratory

gyllen@llnl.gov

Abstract

ZeroFault is a powerful commercial AIX-specific memory-error and leak-detection tool. Although it works well on serial and multi-threaded applications, without the use of a wrapper script, it doesn't work well on parallel MPI applications. This talk will present a brief overview of ZeroFault's features, the details of how to make ZeroFault (and other serial tools) work well at scale with wrapper scripts, and eight tips and tricks that you should know before using ZeroFault on parallel scientific applications.

ZeroFault Features

- Detects a comprehensive set of memory errors
  - Reads/writes of unallocated stack, heap, and static memory
  - Reads/writes of freed heap memory
  - Reads of uninitialized stack, heap, and static memory
  - Attempts to free or realloc unallocated memory
  - Several types of memory leaks
- Works on any existing AIX 32-bit executable
  - Uses run-time emulator, no recompilation or relinking
  - Checks shared libraries and dynamically loaded modules, including third-party libraries (e.g. OS and MPI libraries)
  - Works on all apps (including " tool-killers") tried so far

ZeroFault Limitations

- Significant overhead introduced
  - Applications run 8X-40X slower
    - Need to pick example where error manifests quickly
    - Need to scale up requested batch time appropriately
  - Applications use up to 2X memory
    - May need to use less tasks per node, if memory limited
  - Similar overheads seen with other memory checkers
- Instruction-set dependent
  - 64-bit support not yet implemented (major effort)
  - Update required to support new instructions (e.g. power4)
    - Use -qarch=604, etc. to pick compatible instruction set
      - Applications with “-g -O” and no “-qarch=auto” work fine

ZeroFault Limitations (II)

- Confused by non-IEEE floating point modes
  - Some apps don't want IEEE underflow, etc.
    - Use "magic" code to get a non-IEEE floating point mode
  - Recently discovered to sometimes hang ZeroFault
    - Working on test-case to give to The ZeroFault Group
  - Currently need to disable special mode during testing
- Not a parallel memory tool
  - Issues and workarounds described in this presentation

ZeroFault's Interaction with POE

- For serial (non-MPI) apps, straightforward use
  - Usage: zf [zf_options] serial_app [app_args]
  - ZeroFault checks app and brings up GUI
- For MPI apps, same usage yields unexpected
  - Running "zf mpi_app" checks poe, not mpi_app!
    - All MPI applications call poe automatically during startup
      - If mpi_app not already launched, launches parallel job on remote node(s)
    - ZeroFault appears to work, but no errors will be detected
    - Same issue seen other serial tools (e.g. time, hpmcount)
- Need to use poe to launch zf on MPI apps
  - Running 'poe zf mpi_app' works, but several issues remain
  - Wrapper scripts can be used to resolve these issues

Wrapper Script Considerations

- Usually don't want GUI spawned during zf run
  - Too many GUI windows created
    - One for each MPI task zf run on
    - No easy way to get the MPI rank for the GUI windows
  - Not batch friendly
    - Must wait for batch run to start and setup DISPLAY properly
    - GUI keeps nodes allocated even after app terminates
      - Burns bank time and nodes are idle (E.g., if keep GUI up until morning)
    - Closing any task's ZeroFault GUI will terminates application
  - Use zf option '-d none' to prevent spawning of GUI
    - Use 'zf_ui' on .zfo files to view output with GUI after run
  - Spawned GUI necessary to use memory snapshot features

Wrapper Script Considerations (II)

- Need to organize zf output using MPI rank
  - Environment variable MP_CHILD set by poe to MPI rank
  - ZeroFault output name has fixed form: app_name.n.zfo
    - n starts at 1 and is incremented to avoid naming conflicts
  - Create directory zf_(rank) and put zf output in directory
    - Use zf option '-o zf_$MP_CHILD' to specify output dir
- Need to be able to run zf on subset of tasks
  - May want limit zf output to task or tasks having problems
  - One floating zf license per user per node used
    - On NH2, 4 licenses can cover 64 MPI tasks on 4 nodes
    - Unlimited licenses now available to supercomputer centers
  - Use $MP_CHILD to select tasks to run ZeroFault on

Wrapper Script Considerations (III)

- Most AIX 4.3 shells (/bin/sh, ...) conflict with poe
  - Shells (in some conditions) block signals poe uses
    - Problem typically only occurred with 14+ tasks per node
  - Only optionally installed '/usr/dt/bin/dtksh' works at scale
    - Need to start wrappers with '#!/usr/dt/bin/dtksh'
    - Need to request optional 'dt' package to be installed
- IBM indicated AIX 5.1 shells compatible with poe
  - Have not tested AIX 5.1 shells at scale in wrapper scripts

LLNL's ZeroFault Wrapper Scripts

- zf_id: run ZeroFault on one MPI rank (no GUI)
  - Usage: poe zf_id MPI_ID mpi_app [args]
  - Useful when one task clearly has a problem
- zf_idx: GUI version of zf_id
  - Use interactively or with "batch xterm trick"
  - Normal leak detection finds only memory not pointed to
    - Memory is often not freed before application terminates
    - Leak detection not done if application terminates abnormally
  - Allows use of ZeroFault's memory snapshot features
    - Snapshot lists all memory currently allocated
    - Comparing snapshots shows exactly what changed
      - Usually done between cycles, when all transient memory should be free

LLNL's ZeroFault Wrapper Scripts (II)

- zf_range: run on continuous range of MPI ranks
  - Usage: poe zf_range START_ID END_ID mpi_app [args]
  - When zf licenses limited and problem task rank unknown
- zf_all: run on all tasks
  - Usage: poe zf_all mpi_app [args]
  - Requires unlimited license for large scale runs
- User requested enhancement to wrapper scripts
  - Making 'poe' optional when invoking zf wrapper scripts
    - Sometimes forget poe in batch scripts and not detect until their batch job fails
- Email me at gyllen@llnl.gov for copy of scripts

ZeroFault Installation Tweaks

- Need to expand default ZeroFault filter file
  - Simple MPI programs create over a thousand warnings!
    - From libhal_r.a, libmpi_r.a, libmpic_r.a, libpthreads.a, etc.
    - Almost all safe to ignore (or cannot fix anyway)
    - Overwhelms users and hides relevant messages
  - File to enhance is /usr/lpp/ZeroFault/bin/zf_suppress
    - Use minimal filtering to prevent suppression of real errors
      - I.e. filter for libmpi_r.a includes " UMR function _make_req"
      - Empirically generated 20 or so rules from representative MPI apps
  - Train users to only briefly consider remaining messages
- Need to increase memory limits for zf_ui
  - Binary edit ld header (we use/wrote script setbmaxdata):
    - /usr/bin/echo '\0200\0\0\0' | dd count=1 bs=4 conv=notrunc of=/usr/lpp/ZeroFault/bin/zf_ui seek=19

False Positives: Bad Memory Writes

- For all F90 intrinsics that reduce arrays to scalars
  - MAXVAL(), MAXLOC(), SUM(), etc.
    - E.g.: max = MAXVAL(a(:)) ← ZeroFault flags BMW here
  - ZeroFault fooled by tricky temporary generated by xlf90
- If use realloc() in multithreaded code regions
  - Race condition in ZeroFault's realloc() state update
    - Causes good ptrs returned by realloc, malloc, etc. to look bad
      - Only has been seen when multiple threads call realloc() simultaneously
    - IBM realloc() behavior conflicts with ZeroFault design
      - Workaround: Use custom version of realloc() that uses malloc() and free()
  - Actual errors due to incorrect usage of realloc() more likely!
    - realloc (ptr, new_size); ← ptr invalid if ptr moved by realloc
    - ptr = realloc (ptr, new_size); ← Correct usage of realloc

False Positives: Null Pointer Reads

- Null pointer reads don't segfault on AIX
  - All reads from address 0 to 4k (page 0) return 0
  - This allows memory latency reducing optimizations
    - if (ptr != NULL) { a+= *ptr; } /* Before Optimization */
    - t = *ptr; if (ptr != NULL) {a+=t;} /* After Optimization */
  - Unfortunately hides erroneous reads of NULL pointers
    - Code segfaults if run on different platforms
    - Possible incorrect results due to using returned 0 as data
- ZeroFault warns about only in -g compiled code
  - Uses -g as "unoptimized" code marker (see manual)
  - If compiled with '-g -O', may get many(!) RNULL errors
    - Either ignore or compile with only -g to see if really exist

False Positives: Uninitialized Stack Reads

- Seen frequently in MPI and AIX libraries
  - Documentation states can be caused by optimization
  - Users have seen them in '-g -O' compiled code
    - If recompiling with just '-g' removes, then safe to ignore
    - Zerofault doesn't warn about in '-O' code
  - Sometimes real errors, but often never figured out
- Users often frustrated by this type of error
  - Use filter feature to hide warnings that can be ignored

User Experiences

- Helped find many subtle memory problems
  - Header file mismatches with library versions (struct sizes)
    - Caused by -L/usr/local/lib, etc. grabbing wrong version
  - Buffers read in from restarts not completely filled in
  - Subtle ordering issues when calling C++ destructors
  - Multiple frees of same pointer only when multithreaded
  - Hard-coded "big" limits on buffers, domains, etc.
  - Wide variety of memory leaks
  - Subtle memory growth issues (e.g., expanding hash tables)
- Run extensively at scale (using wrapper scripts)
  - Many problems can only be reproduced at scale (> 1000 MPI tasks) and on the same platform

User Experiences (II)

- Memory tools used extensively
  - Many consider them just as vital as traditional debuggers
  - Want at least one on each supercomputing platform
- Many strange behaviors are not memory errors
  - Function call parameters flipped or wrong
  - APIs or interfaces misinterpreted
  - Subtle logic errors
  - Fixing memory problems detected often doesn't fix app!
- Users usually stressed, want clear answers
  - First time users often wants help interpreting results
  - Frustrated by false positives unless warned about up front

UCRL-PRES-149673

Work performed under the auspices of the U. S. Department of Energy by the University of California, Lawrence Livermore National Laboratory under Contract W-7405-Eng-48